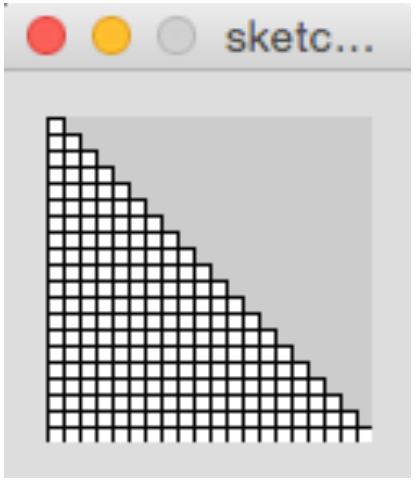
Chapter 7

Functions

Recap

- function call statements
- setup() and draw()
- variables: declaration, use, assignment
- if-else
- loops: while & for
- various operators (arithmetic, boolean, relational, shortcut versions)
- 2D drawing primitives including translate() and rotate()
- comments & readability (variable names, use of white space, few magic numbers)

Write a program that draws this image. The little squares are 5x5. The screen is the default 100x100. It doesn't change.



```
//Draws the grid on the previous slide
int squareWidth;
int numRows;
void setup(){
  size(100,100);
  squareWidth = 5;
  numRows = width/5;
  //noFill();
}
void draw(){
  for(int i = 0;i < numRows;i++){
    for(int j = 0;j <= i;j++){
      rect(j*squareWidth,i*squareWidth,squareWidth,squareWidth);
    }
  }
}
```

```
size(600, 400);
int frontOfCar = 100;
int topOfCar = 100;
int lengthOfCar = 200;
int carBodyHeight = lengthOfCar/4;
int wheelDiameter = lengthOfCar/4;

// draw first car
// draw the body
rect(frontOfCar, topOfCar, lengthOfCar, carBodyHeight);

// draw the wheels
ellipse(frontOfCar+wheelDiameter, topOfCar+carBodyHeight,
wheelDiameter, wheelDiameter);
ellipse(frontOfCar+lengthOfCar-wheelDiameter,
topOfCar+carBodyHeight, wheelDiameter, wheelDiameter);

// draw the windshield
line(frontOfCar+carBodyHeight, topOfCar,
frontOfCar+2*carBodyHeight, topOfCar-carBodyHeight);
```

```
// draw the second car
frontOfCar = 300;
topOfCar = 300;
lengthOfCar = 150;
carBodyHeight = lengthOfCar/4;
wheelDiameter = lengthOfCar/4;

// draw the body
rect(frontOfCar, topOfCar, lengthOfCar, carBodyHeight);

// draw the wheels
ellipse(frontOfCar+wheelDiameter, topOfCar+carBodyHeight,
    wheelDiameter, wheelDiameter);
ellipse(frontOfCar+lengthOfCar-wheelDiameter,
    topOfCar+carBodyHeight, wheelDiameter, wheelDiameter);

// draw the windshield
line(frontOfCar+carBodyHeight, topOfCar,
    frontOfCar+2*carBodyHeight, topOfCar-carBodyHeight);
```

```
// draw the third car
frontOfCar = 350;
topOfCar = 100;
lengthOfCar = 100;
carBodyHeight = lengthOfCar/4;
wheelDiameter = lengthOfCar/4;

// draw the body
rect(frontOfCar, topOfCar, lengthOfCar, carBodyHeight);

// draw the wheels
ellipse(frontOfCar+wheelDiameter, topOfCar+carBodyHeight,
    wheelDiameter, wheelDiameter);
ellipse(frontOfCar+lengthOfCar-wheelDiameter,
    topOfCar+carBodyHeight, wheelDiameter, wheelDiameter);

// draw the windshield
line(frontOfCar+carBodyHeight, topOfCar,
    frontOfCar+2*carBodyHeight, topOfCar-carBodyHeight);
```

There must be a better way

- What if I decide to modify the look of the car a bit? I have to make many changes in 3 places.
- What if I want yet another car? I have to copy and paste a big chunk of code.
- You have to scan a moderate amount of code to determine I’m drawing 3 cars.

```
void setup() {
    size(600, 400);
    drawCar(100, 100, 200);
    drawCar(300, 300, 150);
    drawCar(350, 100, 100);
}
void drawCar(int frontOfCar, int topOfCar, int lengthOfCar) {
    int carBodyHeight = lengthOfCar/4;
    int wheelDiameter = lengthOfCar/4;

    // draw the body
    rect(frontOfCar, topOfCar, lengthOfCar, carBodyHeight);
    ...
```

```
void drawCar(int frontOfCar, int topOfCar, int lengthOfCar) {
    int carBodyHeight = lengthOfCar/4;
    int wheelDiameter = lengthOfCar/4;

    // draw the body
    rect(frontOfCar, topOfCar, lengthOfCar, carBodyHeight);

    // draw the wheels
    ellipse(frontOfCar+wheelDiameter, topOfCar+carBodyHeight,
        wheelDiameter, wheelDiameter);
    ellipse(frontOfCar+lengthOfCar-wheelDiameter,
        topOfCar+carBodyHeight, wheelDiameter, wheelDiameter);

    // draw the windshield
    // use carBodyHeight to also control the size of the windshield
    line(frontOfCar+carBodyHeight, topOfCar,
        frontOfCar+2*carBodyHeight, topOfCar-carBodyHeight);
}
```

```
// A very simple function to draw a square
void setup(){
   size(400,400);
}

void draw(){
   simpleSquare();
}

void simpleSquare(){
   line(0,0,0,20);
   line(0,20,20,20);
   line(20,20,20,0);
   line(20,0,0,0);
}
```

```
// A very simple function to draw a square
void setup(){
   size(400,400);
}

void draw(){
   simpleSquare();
}

void simpleSquare(){
   rect(0,0,20,20);
}
```

Defining Simple Methods

ReturnType Identifier (ParameterList) {

Body

}

- ReturnType is the type of value returned from the method/function.
- Identifier is the name of the method/function.
- ParameterList is a list of variables that will be used to pass information into the method. These are called the formal parameters.
- Body is a list of statements and declarations describing the action performed by this method.

```
// carAndHouse
void setup() {
    size(600,400);
    translate(10,height/2);
    drawCar();
    translate(width/2, 0);
    drawHouse();
}

/* Draws a car with the top front at (0,0).
Use translate() to position the car before calling drawCar().
*/
void drawCar() {
    int carWidth = 40, carHeight = carWidth/2;
    fill(255,0,0);
    rect(0, 0, carWidth, carHeight);
    fill(0);
    ellipse(carWidth/4, carHeight, carHeight/2, carHeight/2);
    ellipse(carWidth*3/4, carHeight, carHeight/2, carHeight/2);
}
```

```
/*
Draw a house with the middle of the house baseline at (0,0).
*/
void drawHouse() {
    int houseWidth = 250;
    int houseHeight = houseWidth/3;

    // draw house
    fill(#6F3506); // brown house
    noStroke();
    rect(-houseWidth/2, -houseHeight, houseWidth, houseHeight);
}
```

```
//How many flowers does this program draw?
void setup() {
    size(600, 400);
}
void flower(int x, int y) {
    pushMatrix(); // needed because we use translate/rotate
    stroke(255, 255, 0);
    fill(100, 0, 100);
    translate(x, y); // move to the center of the flower
    int numPetals = 16;
    int petalLength = 60, petalWidth = petalLength/3;
    float angle = 2*PI/numPetals;
    for (int i = 0; i < numPetals; i = i + 1) {
        ellipse(0, petalLength/2, petalWidth, petalLength);
        rotate(angle);
    }
    popMatrix(); // put things back the way they were
}
```

A. 0
B. 1
C. 16

```
void setup() {
    size(600,400);
    translate(10,height/2);
    drawCar();
    translate(width/2, 0);
    drawHouse();
    translate(width/4, 0);
    drawCar();
}

/* Draws a car with the top front at (0,0).
Use translate() to position the car before calling drawCar().
*/
void drawCar() {
    int carWidth = 40, carHeight = carWidth/2;
    fill(255,0,0);
    rect(0, 0, carWidth, carHeight);
    fill(0);
    ellipse(carWidth/4, carHeight, carHeight/2, carHeight/2);
    ellipse(carWidth*3/4, carHeight, carHeight/2, carHeight/2);
}
```

```
// carGlobalParams
void setup() {
    size(600,400);
    carX = 10;
    carY = height/2;
    carWidth = 40;
    drawCar();
    carX = width/2;
    carY = height/4;
    carWidth = 20;
    drawCar();
}
```

```
...
    carX = width/2;
    carY = height/4;
    carWidth = 20;
    drawCar();
}
int carWidth, carX, carY;
void drawCar() {
    int carHeight = carWidth/2;
    fill(255,0,0);
    rect(carX, carY, carWidth, carHeight);
    fill(0);
    ellipse(carX + carWidth/4, carY + carHeight,
    carHeight/2, carHeight/2);
    ellipse(carX + carWidth*3/4, carY + carHeight,
    carHeight/2, carHeight/2);
}
```

```
// carFormalParams
void setup() {
    size(600,400);

    drawCar(10, height/2, 40);
    drawCar(width/2, height/4, 20);
}
void drawCar(int carX, int carY, int carWidth) {
    int carHeight = carWidth/2;
    fill(255,0,0);
    rect(carX, carY, carWidth, carHeight);
    fill(0);
    ellipse(carX + carWidth/4, carY + carHeight,
            carHeight/2, carHeight/2);
    ellipse(carX + carWidth*3/4, carY + carHeight,
            carHeight/2, carHeight/2);
}
```

```
void setup() {
    size(400, 400);
}
void draw() {
    background(150);

    drawCar(50, 50, 40);
    drawCar(mouseX, mouseY, (mouseX+mouseY)/10);
}
```

Three versions of drawCar()

```
draw the car at 0,0 then use translate
translate(100, 200);
drawCar();

draw the car at carX, carY
carX = 100;
carY = 200;
drawCar();

draw the car by passing parameters to a method
drawCar(100, 200);
```

Which is not true of the 3rd version – drawCar(100,200)?

A. It takes less typing for each new car.

B. The user doesn't need to know the name of any global variables.

C. It doesn't mess with the current coordinate system (rotations and translations).

D. The definition itself is shorter.

E. You never have to hunt around to figure out where the car is being drawn.

Create a function square() that takes three parameters, the x and y coordinates of the upper left corner of the square and the length of the sides of the square. The function should draw the specified square.

```
/* program that uses a function to draw a square.
*/
void setup(){
  size(400,400);
}

void draw(){
  // Use square function that was created below.
  // It is OK that the function is being called
  // above where the function is defined.
  square(20,20,40);
  square(40,40,80);
}

// create square function
void square(int x, int y, int size){
  rect(x,y,size,size);
}
```

```
/* This program demonstrates the how values are passed to a function.
The parameters take on the value that is passed to them during the
function call.
*/
void setup(){
  int a = 50;
  int b = 51;
  int c = 52;

  // call printParameters function with different types of input
  printParameters(1, 5, 9);
  printParameters(a, b, c);
  printParameters(a+b, a+c, b+c);
  printParameters(b%a, c%a, a%c);
}

// this is a function that takes in 3 integer parameters (x,y,z)
void printParameters(int x, int y, int z){
  println("Value of x: " + x +
          ", value of y: " + y +
          ", value of z: " + z);
}
```

```
// twoButtons
int buttonWidth, buttonHeight;
void setup() {
    size(400,400);
    background(255); // 255 means white

    // make button size proportional to the display size
    buttonWidth = width/3;
    buttonHeight = height/10;
    int buttonX, buttonY;
    String label;
```

```
  // draw the first button
  buttonX = 10;
  buttonY = 200;
  label = "one";
  fill(255); // white except for the label below
  rect(buttonX, buttonY, buttonWidth, buttonHeight);
  fill(255,0,0); // Red text
  text(label, buttonX*1.1, buttonY + buttonHeight*.9);

  // draw the second button
  buttonX = 10;
  buttonY = 300;
  label = "two";
  fill(255); // white except for the label below
  rect(buttonX, buttonY, buttonWidth, buttonHeight);
  fill(255,0,0); // Red text
  text(label, buttonX*1.1, buttonY + buttonHeight*.9);
}
```

Create the drawButton() method needed to make this modification of the previous program work.

```
int buttonWidth, buttonHeight;
void setup() {
   size(400,400);
   background(255); // 255 means white

   // make button size proportional to the display size
   buttonWidth = width/3;
   buttonHeight = height/10;
   drawButton("one", 10, 200, buttonWidth, buttonHeight);
   drawButton("two", 10, 300, buttonWidth, buttonHeight);
}
```

Create a function circle() that takes three parameters, the x and y coordinates of the center of the circle and the radius of the circle. The function should draw the specified circle.

Create this LED Timer

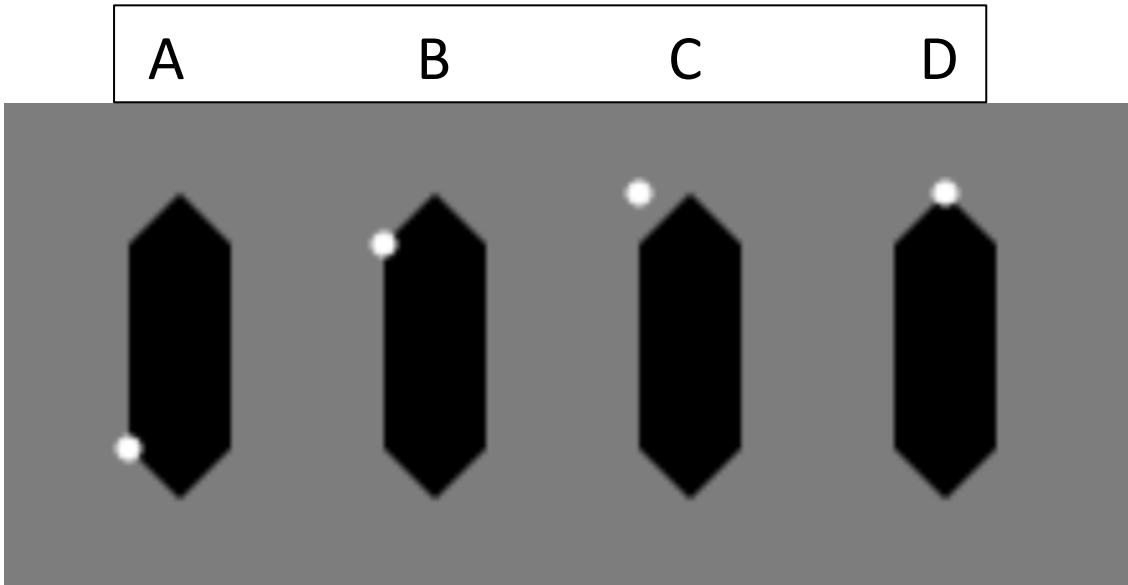
- Draw digital timer elements
- Assemble elements into digits
- Light digit segments to create numbers
- Select number based on a digit

(Derived from an example by Larry Snyder.)

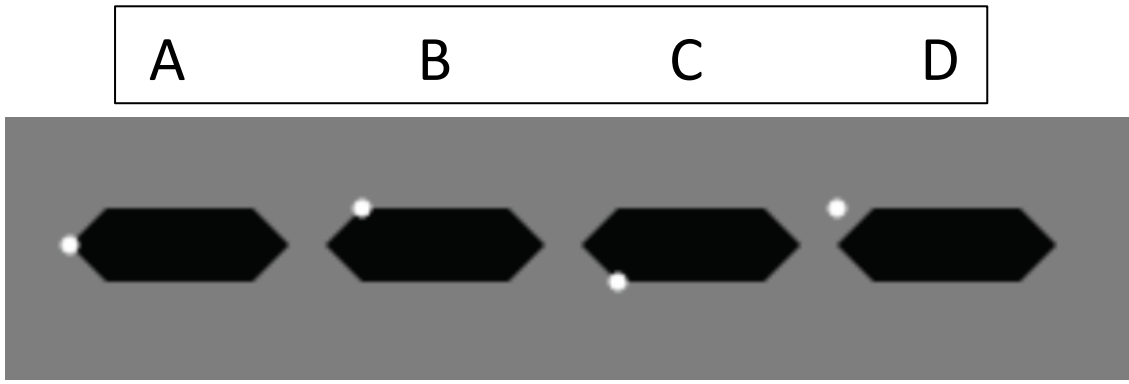
```
void hexa(float xbase, float ybase) {
    rect(xbase, ybase-40, 20, 40);
    triangle(xbase, ybase,
    xbase+20, ybase, xbase+10, ybase+10);
    triangle(xbase, ybase-40,
    xbase+20, ybase-40, xbase+10, ybase-50);
}

void rexa(float xbase, float ybase) {
    triangle(xbase, ybase,
    xbase+10, ybase-10, xbase+10, ybase+10);
    rect(xbase+10, ybase-10, 40, 20);
    triangle(xbase+50, ybase-10,
    xbase+50, ybase+10, xbase+60, ybase);
}
```

```
void test1() {
    background(150);
    int x = 100, y = 100;
    fill(0);
    hexa(100, 100);
    rexa(100,160);
}
```

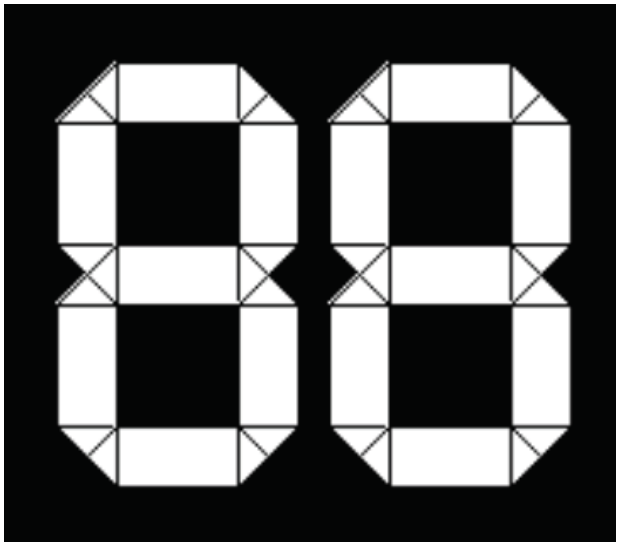
```
void hexa(float xbase, float ybase) {
    rect(xbase, ybase-40, 20, 40);
    triangle(xbase, ybase,
             xbase+20, ybase, xbase+10, ybase+10);
    triangle(xbase, ybase-40,
             xbase+20, ybase-40, xbase+10, ybase-50);
}

void setup() {
    ...
    hexa(x, y);
    fill(255);
    ellipse(x, y, 5, 5);
}
```



```
void rexa(float xbase, float ybase) {
    triangle(xbase, ybase,
             xbase+10, ybase-10, xbase+10, ybase+10);
    rect(xbase+10, ybase-10, 40, 20);
    triangle(xbase+50, ybase-10,
             xbase+50, ybase+10, xbase+60, ybase);
}
void setup() {
...
    rexa(x, y);
    fill(255);
    ellipse(x, y, 5, 5);
}
```



Use H'gons to Form A Digit: digit re-uses

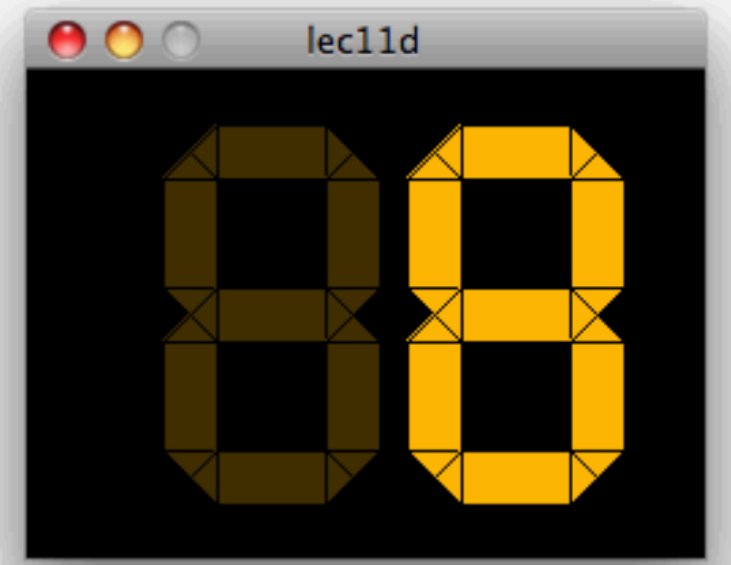


```
// draw all elements with current fill
void digit(float xbase, float ybase) {
    hexa(xbase, ybase); //left upper
    hexa(xbase, ybase+60); //left lower
    rexa(xbase+10, ybase+10); //mid horizontal
    rexa(xbase+10, ybase-50); //top horizontal
    rexa(xbase+10, ybase+70); //bot horizontal
    hexa(xbase+60, ybase); //right upper
    hexa(xbase+60, ybase+60); //right lower
}
```

Let there be light (and dark)

- Define the illumination of the digit
  - Must declare two color variables, initialize to proper colors, use them in fill, and then check them

```
color dark, lite;

void setup( ) {
  size(250, 180);
  background(0);
  stroke(0);
}

void draw( ) {
  lite = color(255,185,0);
  dark = color(64, 48, 0);

  fill(dark);
  digit(50, 20);
  fill(lite);
  digit(140, 20);
}
```



Count In Lights: a function for each number

- Light up the digit for each number:

```
void digit(float xbase, float ybase) {
  hexa(xbase, ybase+10);        //left upper
  hexa(xbase, ybase+70);        //left lower
  rexa(xbase+10, ybase);        //top horizontal
  rexa(xbase+10, ybase+60);     //mid horizontal
  rexa(xbase+10, ybase+120);    //bot horizontal
  hexa(xbase+60, ybase+10);     //right upper
  hexa(xbase+60, ybase+70);     //right lower
}

void one (float xbase, float ybase) {
  hexa(xbase+60, ybase+10);     //right upper
  hexa(xbase+60, ybase+70);     //right lower
}

void two (float xbase, float ybase) {
  rexa(xbase+10, ybase);        //top horizontal
  rexa(xbase+10, ybase+60);     //mid horizontal
  rexa(xbase+10, ybase+120);    //bot horizontal
  hexa(xbase+60, ybase+10);     //right upper
  hexa(xbase, ybase+70);        //left lower
}
```

Select A Number To Display

- Given an integer, display it in lights

```
void sel(int n, float xbase, float ybase) {
  fill(lite);
  if (n == 0) {
      zero(xbase, ybase);
  }
  if (n==1) {
      one(xbase, ybase);
  }
  if (n==2) {
      two(xbase, ybase);
  }
  if (n==3) {
      three(xbase, ybase);
  }
  if (n==4) {
      four(xbase, ybase);
  }
  if (n==5) {
      five(xbase, ybase);
  }
  if (n==6) {
      six(xbase, ybase);
  }
      ...
```

Create a 3 Digit Display

```
void three_digit(int n, float xbase, float ybase) {
  fill(dark);
  digit(50,90);
  digit(140, 90);
  digit(260, 90);
  fill(lite);
  rect(xbase+185, ybase+125, 15, 15);
  sel((n/100)%10, xbase, ybase);
  sel((n/10)%10, xbase+90, ybase);
  sel(n%10, xbase+210, ybase);
}
```

Here's The Action

What is the value of 137%10?

A. 7

B. 13

C. 13.7

D. 14

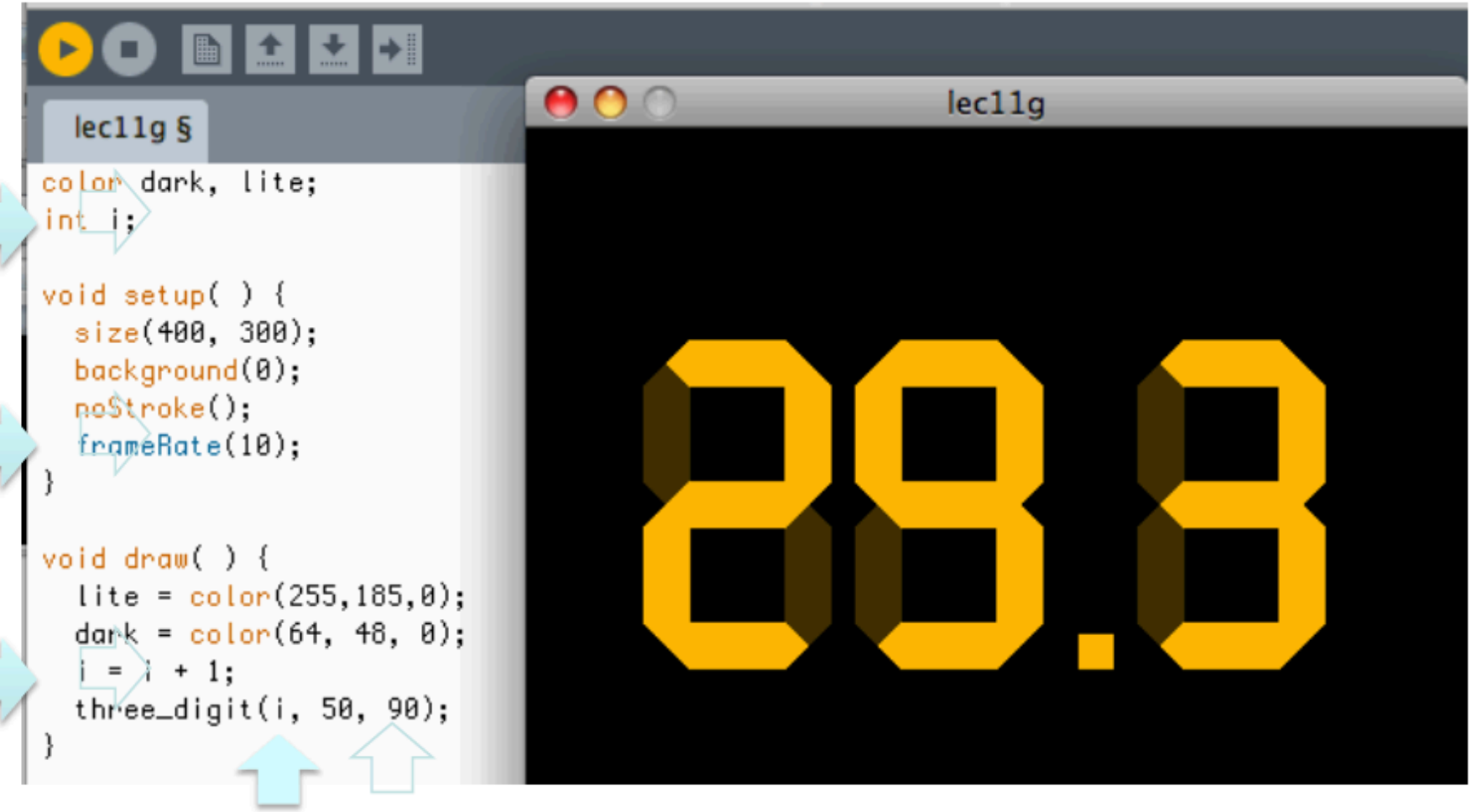
What is the value of (137/10)%10?

A. 1

B. 3

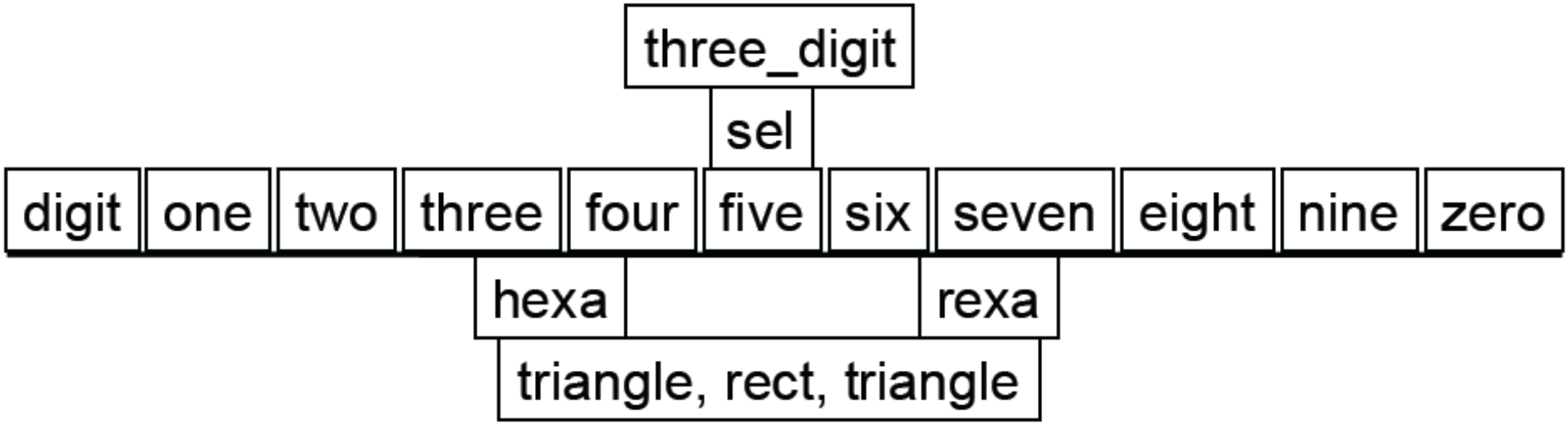
C. 7

Count up At The Frame Rate



Functional Abstraction: Layers of Functions

- Review What We Did



- The computation is ONLY drawing triangles and rectangles, but we don't think of it that way … to us, it's a timerd

```
void setup() {
    int x = 3;
    it = x;
    printIt();
    it = 10;
    printIt();
    printIt();
}
int it;
void printIt() {
    println(it);
    it = 0;
}
```

```
void setup() {
    int x = 3;
    printVal(x);
    printVal(10);
    println(x);
}
void printVal(int x) {
    println(x);
    x = 0;
}
```

A. They both print 3, 10, 0 on separate lines
B. They both print 3, 10, 3 on separate lines
C. The left prints (A) and the right prints (B)
D. The left prints (B) and the right prints (A)
E. I have no idea